EE290O Project

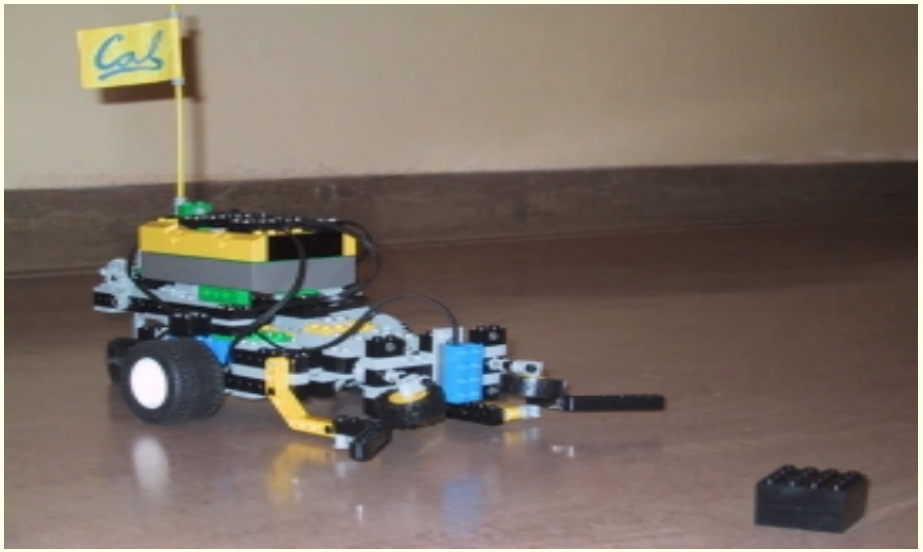
Yang Zhao

Arkadeb Ghosal

May 14, 2002

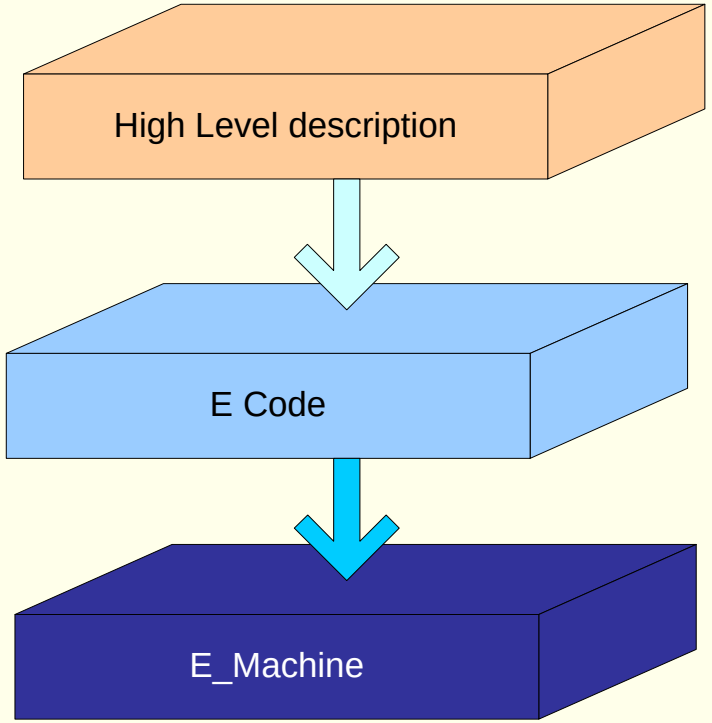
Demo I

- A rectangular playground
- Robot scans the field
- If any object is found it picks it up and returns to the starting point

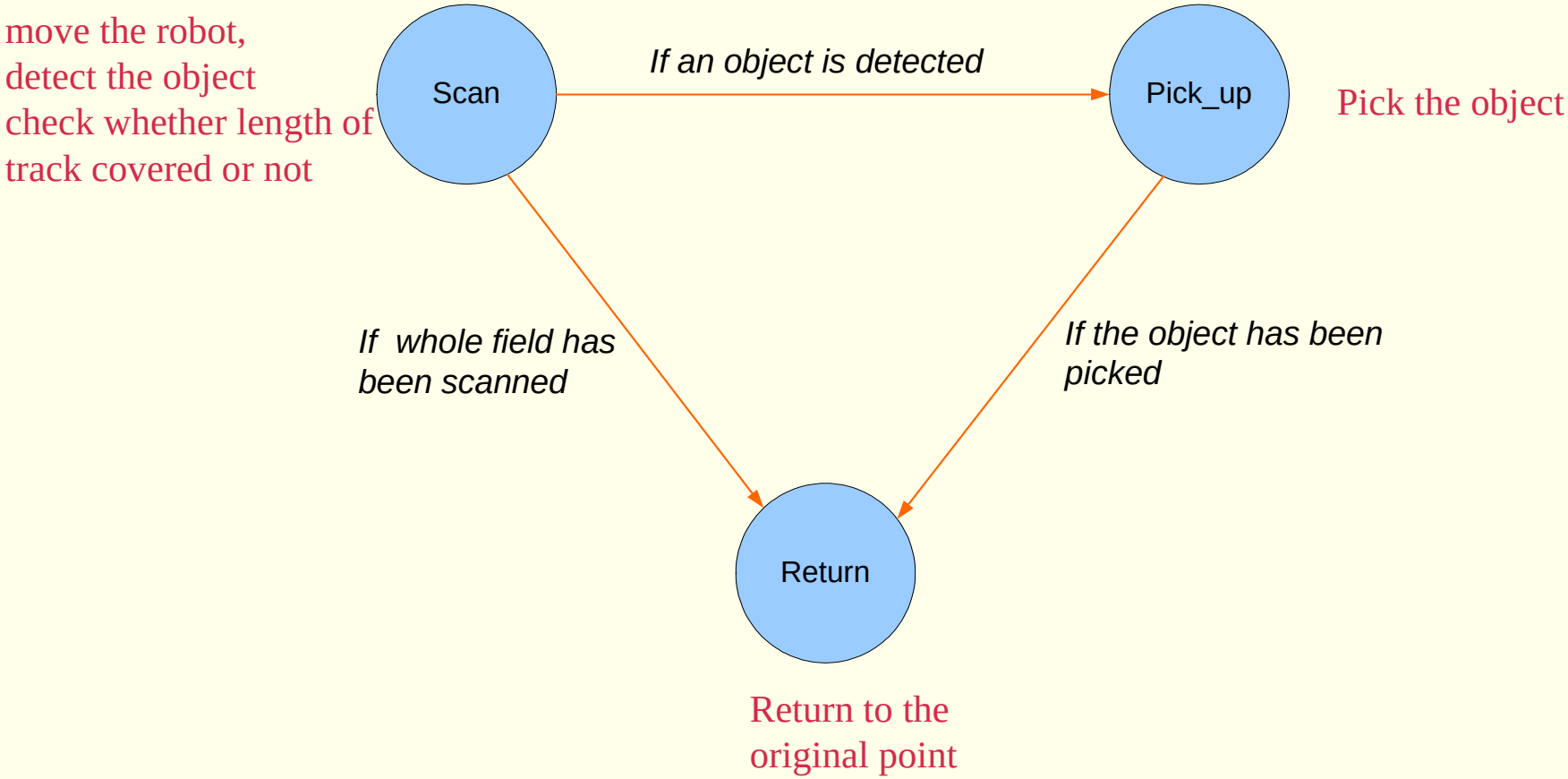


Implementation strategy

- Design from a high-level
- Guard the implementation by real-time programming concepts introduced in this course.
  - Use Giotto concept: tasks, drivers, triggers, modes, to describe our design
- Implement the application by e-code based on our E machine

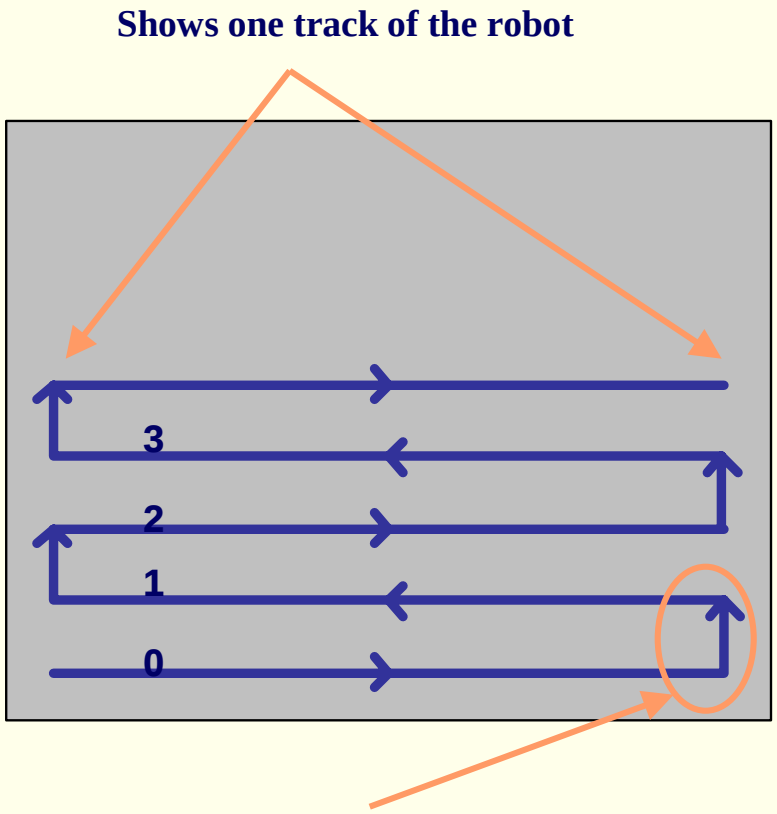


High Level Description

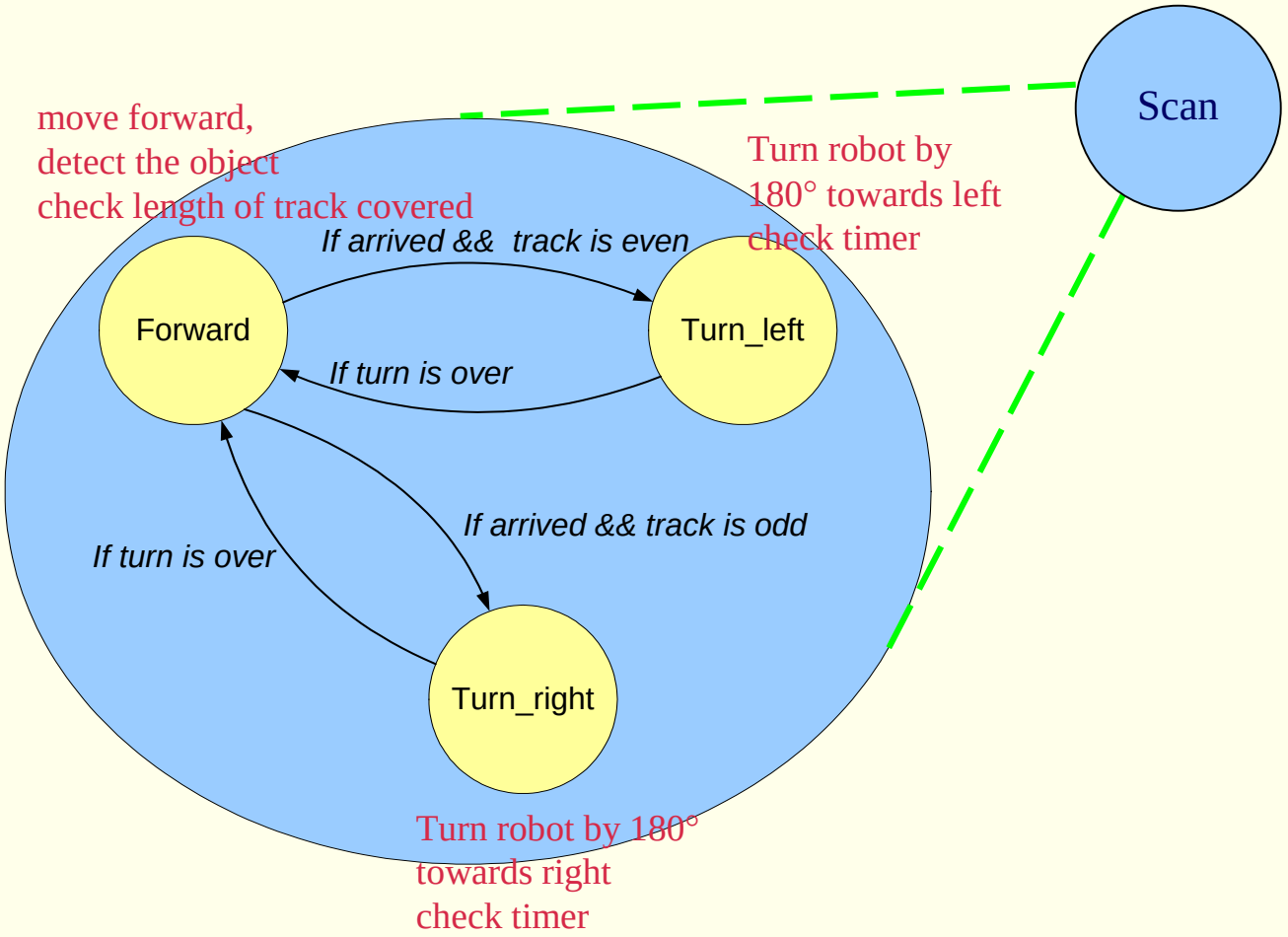


Our Scan Algorithm

- Scanning technique is very simple
  - Not the best but it serves our purpose
  - The robot covers a pre-specified distance called the track
  - If it detects any object on the track it goes to the pick up task.
  - Otherwise on completion of an even track, it takes an left U-turn and continue.
  - On comletion an odd track, it takes an right U-turn and continue.

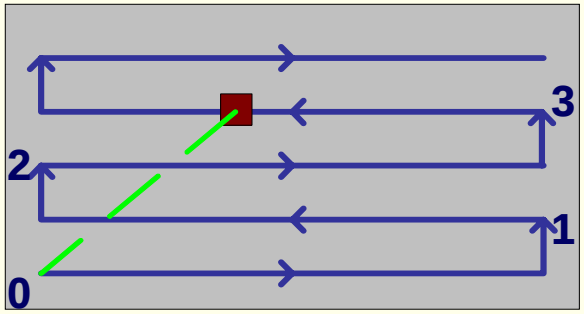


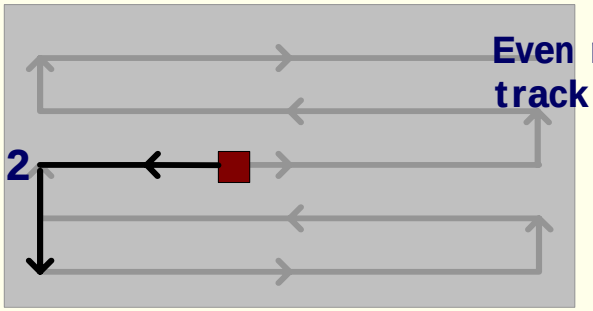
Refine the Scan task

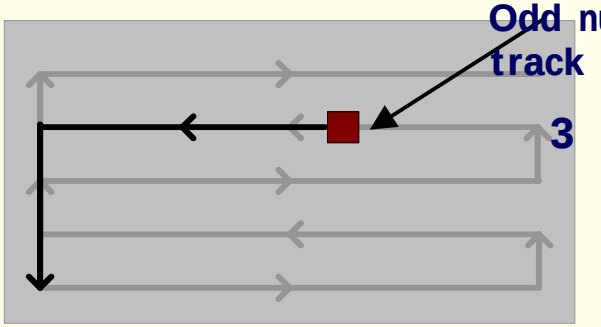


Return Algorithm

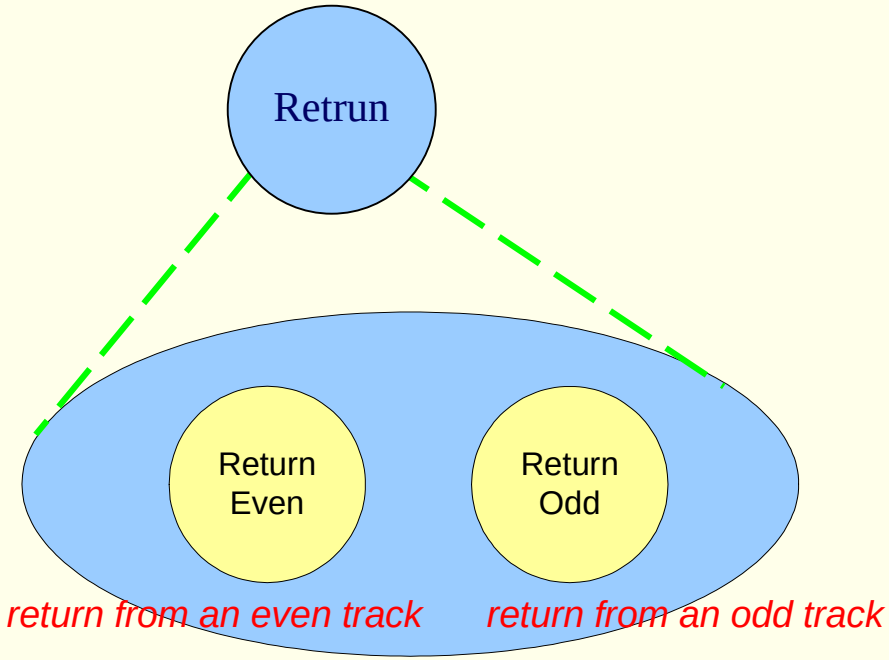
- Ideally, should take the shortest straight line distance
  - There isn't suitable math function in LegOS to get the angle
- Approximate the ideal return..
  - If the object is at an even numbered track, reverse, turn right by 90° and move to the origin
  - If the object is at an odd numbered track, go forward, turn left by 90° and move to the origin



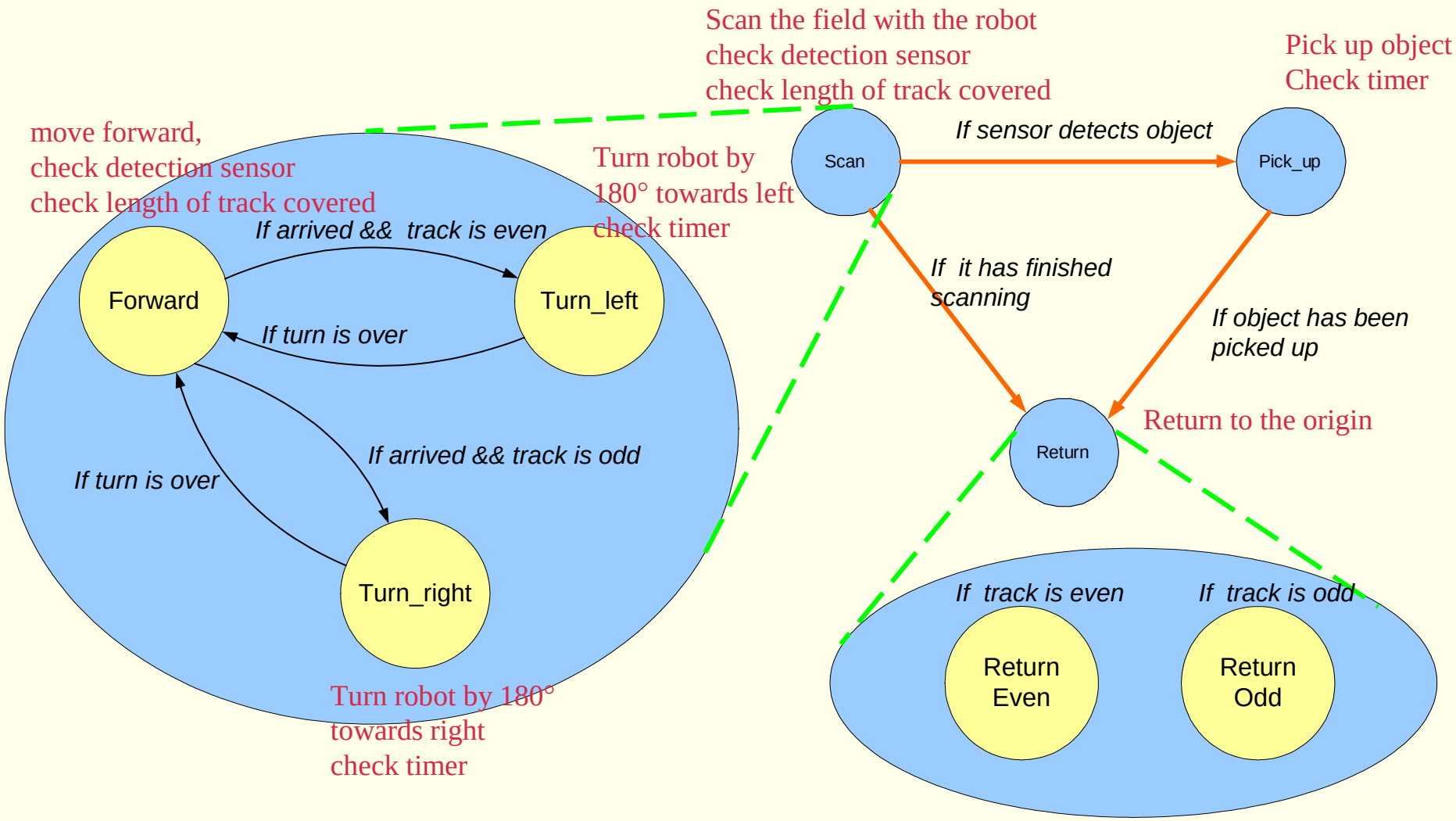




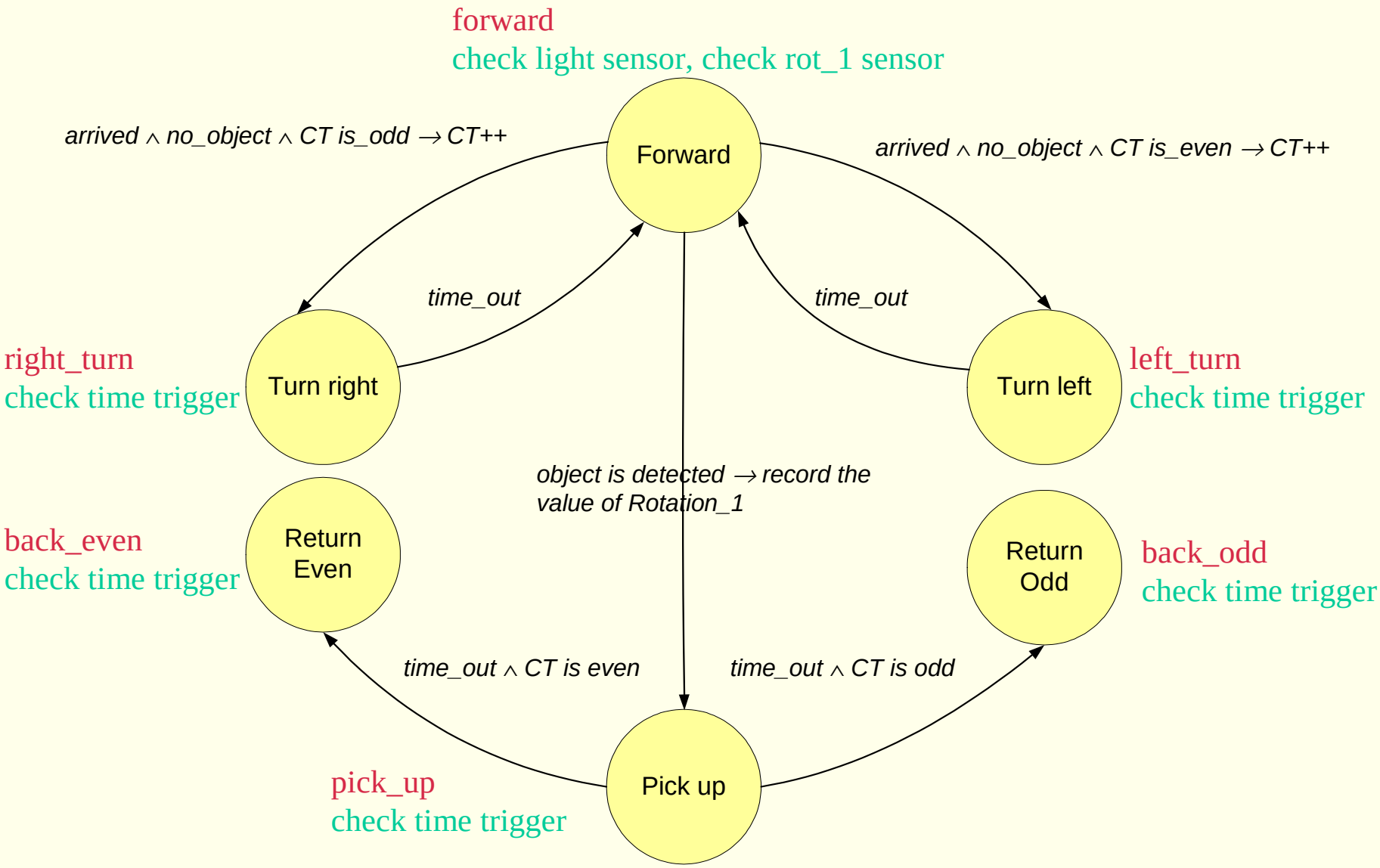
Refine the Return task



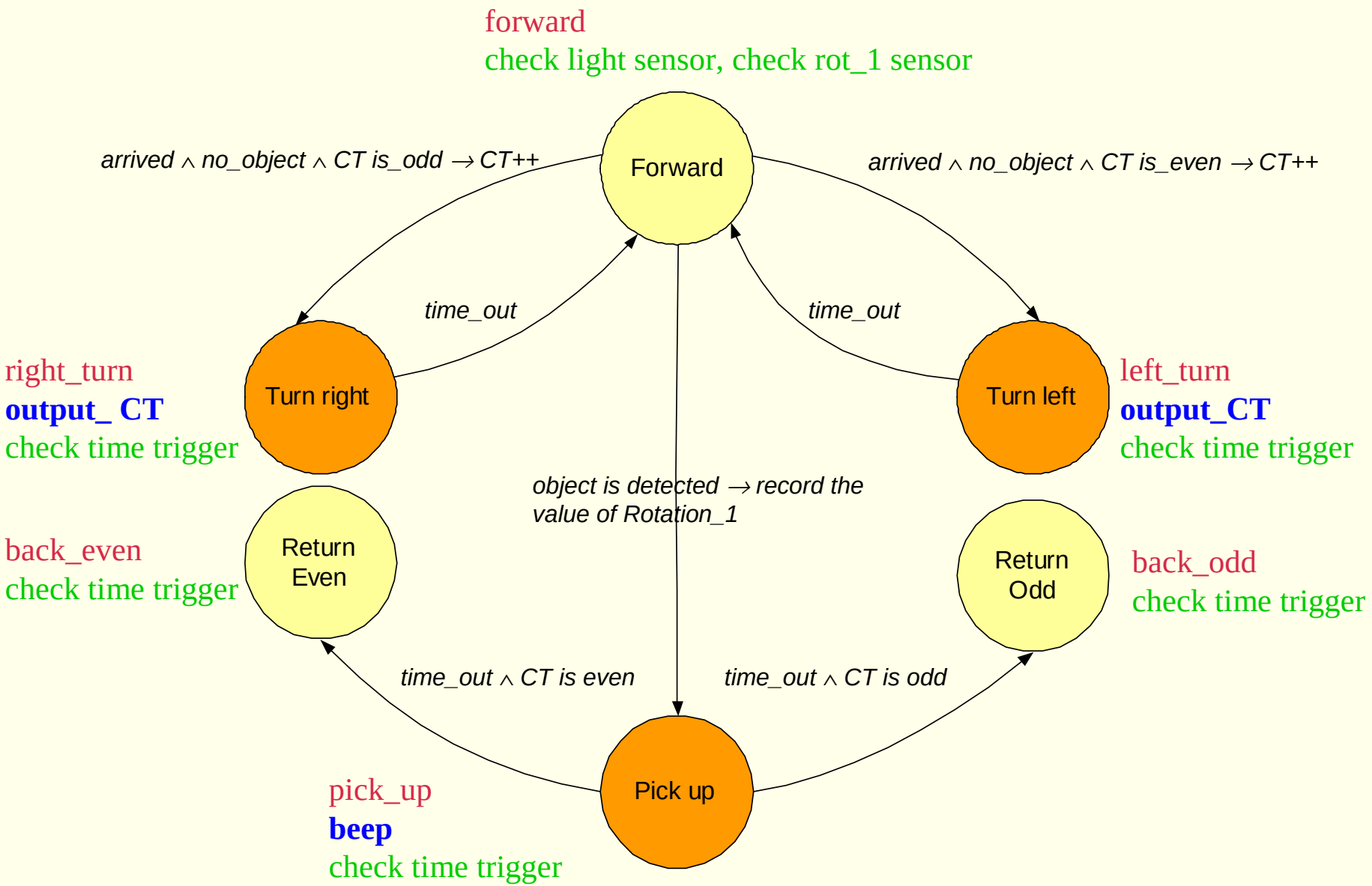
A closer look



The Giotto conceptualization



Adding concurrent tasks



What do we use ?

- motors
  - motor a -- move forward
  - motor b -- the radar motor
    - combining motor a,b robot can take any turn
  - motor c -- picks the object
- sensors
  - rotation sensors
    - rot_1, counts the no. of rotations taken by motor a
    - rot_2, does the same thing for motor b
  - light sensor
    - detects object of pre-specified darkness
- object
  - a black plastic cuboid of approx size 4cm X 1 cm

The Giotto Code

- sensor
  light_sensor uses read_light_sensor
  rot_1_sensor uses read_rot1_sensor
  rot_2_sensor uses read_rot2_sensor

- //task declarations
  task forward .. schedule c_forward()
  task pick .. schedule c_pick()
  task beep .. schedule c_beep()
  task show .. schedule c_show()
  task left_turn .. schedule c_left_turn()
  task right_turn .. schedule c_right_turn()
  task back_even .. schedule c_back_even()
  task back_odd .. schedule c_back_odd()

- //mode_switch driver
  driver got_a_block
  .. if dark_true(light_sensor) then switch
  driver arrived_even
  .. if arrived_even_true (counter) then switch
  driver arrived_odd
  .. if arrived_even_true (counter) then switch
  driver is_even
  .. if track_is_even_true (counter) then switch;

- mode mode_forward
  //checks whether got_a_block or track over
  //invokes task forward

- mode got_a_block
  //checks whether ready to go back by time
  //invokes pick up
  //invokes beep

- mode go_back_odd
  //performs the return to origin task from
  //an odd_numbered track

- mode go_back_even
  //performs the return to origin task from
  //an even_numbered track

- mode reg_turn_left
  //checks whether to switch to mode_forward
  //invokes U-turn in left direction
  //LCD display of how many tracks has been covered

- mode reg_turn_right
  //checks whether to switch to mode_forward
  //invokes U-turn in left direction
  //display of how many tracks has been covered

The E code

- LS TS RS_1 RS_2 CT

- DARK FWD TURN_180 TURN_90 ROT_LEFT GAP

- void read_light_sensor //writes to LS
- void read_touch_sensor //writes to TS
- void read_rot1_sensor //writes to RS_1
- void read_rot2_sensor //writes to RS_2

- void write_count
  // read CT
  // increments CT and write back

- void set_rot1_sensor //sets SENSOR_1
- void set_rot2_sensor // sets SENSOR_2

- void stop_a // stop motor a
- void stop_b, stop_c

- void forward
  // moving motor a in fwd dir

- void pick // moving motor c forward
- void drop // moving motor c in reverse

- void beep
  //play sound

- int left_turn
  //move b fwd ; move a fwd; move b in rev

- int right_turn ...

- void back_even
  //read RS_1; read CT; calculate the Gap;;
  //go bwd //90 degree right turn //go down

- void back_odd ...

- void output_CT
  // reads CT and writes on LCD

- int arrived_even
  //reads CT & RS_1
  //checks whether CT is even and RS_1 has the req value

- int arrived_odd ...

- int is_even
  //reads counter value
  //checks whether it is odd or even

- int dark
  //reads LS
  //checks light sensor for prespeified darkness value

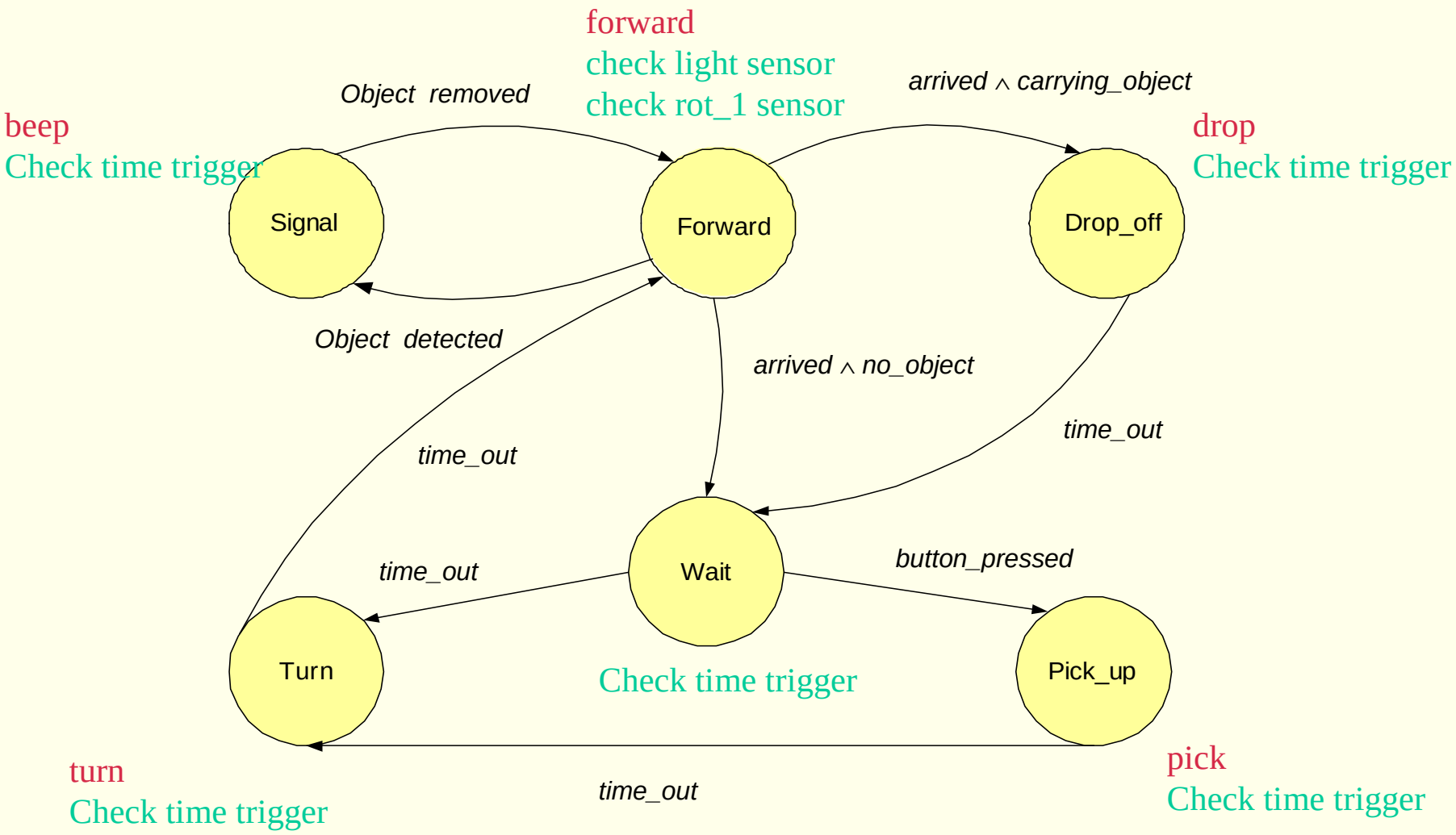
The E Code (cont.)

```
// the forward mode
CALL: set_rot1_sensor;
CALL: read_light_sensor;
CALL: read_rot1_sensor;

COND: arrived_even;
//if not arrived_even check whether arrived_odd
//if arrived_even, go to left_turn

COND: arrived_odd;
//if not arrived_odd resume going forward
//if arrived_odd, go to right_turn

SCHEDULE: forward;

FUTURE: time_trigger
//check sensor

//check dark sensor
COND: dark;
// if not dark continue moving FWD
// if dark pick up

//stop and pick up
CALL: read_rot1_sensor;
CALL: stop_a;

FUTURE: time_trigger;
//start to return

SCHEDULE: pick;

//left_turn mode after forward
CALL: stop_a;
CALL: write_count;

FUTURE: time_trigger;
//start moving forward

SCHEDULE: left_turn;

//right_turn mode after forward
CALL: stop_a;
CALL: write_count;

FUTURE: time_trigger;
//start moving forward

SCHEDULE: right_turn;

//stop mode_2 after pick up
CALL: stop_c;

COND: is_even;
//goto odd_return else goto even_return

SCHEDULE: back_even;

SCHEDULE: back_odd;
```

Demo II

- work as a to-and-fro carrier
  - move in between two fixed points
  - pick up an object from one point and drop it off at another
  - On reaching a point it waits for some pre-specified time and then turns back and returns to other point
  - If an object is need to be send it should be kept in front of the robot and it would pick it up on signaling
    - reaching the other point it drops off the object
  - If it is blocked during moving it signals to move away and stops
    - it resumes the motion once the obstacle has been moved

The modes


forward
check light sensor
check rot_1 sensor
beep
Check time trigger
drop
Check time trigger
turn
Check time trigger
pick
Check time trigger
Signal
Forward
Drop_off
Wait
Turn
Pick_up
Object removed
arrived ∧ carrying_object
Object detected
arrived ∧ no_object
time_out
time_out
time_out
button_pressed
time_out
Check time trigger

Conclusion

- Things learned
  - Real experience about real-time programming.
    - not only software programming
    - mechanical problems
    - light sensor
    - slight deviation can cause a lot of problems
  - Understand and appreciate concepts introduced in this course more deeply.
- Future dirction
  - Feedback
  - How to do efficient scanning
  - Works together with Ptolemy